

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-057329
(43)Date of publication of application : 25.02.2000

(51)Int.Cl.

G06T 1/20
G06F 15/16

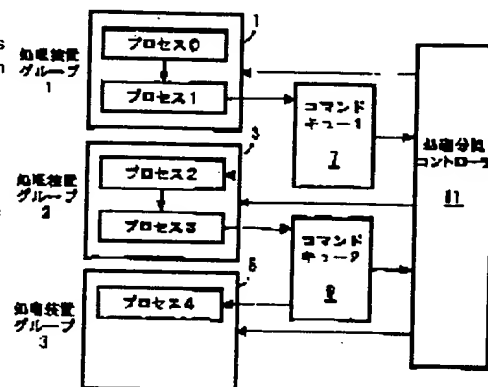
(21)Application number : 10-218852
(22)Date of filing : 03.08.1998

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>
(72)Inventor : KAWASE KATSURA
MORIYAMA TAKAO
NAKAMURA HIDEFUMI

(54) METHOD FOR DYNAMICALLY CHANGING PROCESSING ALLOTMENT AND COMPUTER

(57)Abstract:

PROBLEM TO BE SOLVED: To draw out the performance of a whole system as much as possible by detecting the increase/decrease of a feature variable in a queue for transmitting/receiving processed results between plural groups and changing the allotment of processing in the respective groups based on the increase/ decrease of the feature variable.
SOLUTION: A processing allotment controller 11 monitors command queues 1 (7), 2 (9) and detects the increase/decrease of a feature variable to be an index in the case of changing the processing allotment. When the feature variable of the command queue 1 (7) is reduced to more than a prescribed threshold, the controller 11 instructs a processor group 1 (1) to execute only a process 0 and instructs a processor group 2 (3) also to execute a process 1. When the feature variable of the command queue 2 (9) is reduced to more than the prescribed threshold, the allotment processes are adjusted between both processor groups 2 (3) and 3 (5).



LEGAL STATUS

[Date of request for examination] 27.12.1999
[Date of sending the examiner's decision of rejection] 27.02.2001
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number] 3224782
[Date of registration] 24.08.2001
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2000-57329

(P2000-57329A)

(43) 公開日 平成12年2月25日 (2000.2.25)

(51) Int.Cl.

識別記号

F I

テマコード(参考)

G 0 6 T 1/20

G 0 6 F 15/66

L 5 B 0 4 5

G 0 6 F 15/16

3 7 0

15/16

3 7 0 N 5 B 0 5 7

審査請求 未請求 請求項の数12 O L (全10頁)

(21) 出願番号

特願平10-218852

(22) 出願日

平成10年8月3日 (1998.8.3)

(71) 出願人 390009531

インターナショナル・ビジネス・マシー
ズ・コーポレーション

INTERNATIONAL BUSIN
ESS MASCHINES CORPO
RATION

アメリカ合衆国10504、ニューヨーク州

アーモンク (番地なし)

(74) 代理人 100086243

弁理士 坂口 博 (外1名)

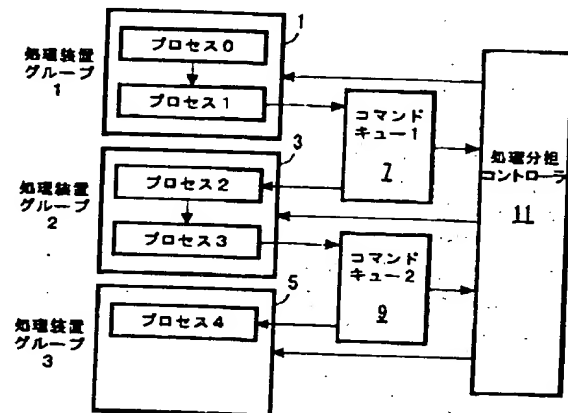
最終頁に続く

(54) 【発明の名称】 処理分担動的変更方法及びコンピュータ

(57) 【要約】

【課題】 処理量及び処理能力が予測不可能なマルチ・プロセッシング・システムにおいて、処理分担を動的に変更する。

【解決手段】 少なくとも2つのグループに分けることができる複数の処理装置を有するコンピュータにおいて、各グループにおける、第1段から第n段までの一連の処理の分担を動的に変更する本発明は、グループ間の処理結果の受渡しのためのキューにおける特徴量の増減を検出するステップと、特徴量の増減に基づき、各グループにおける処理の分担を変更する変更ステップとを含む。キューに格納されたデータの特徴量は、処理量に関する値を表すものであり、この特徴量を参照しつつ処理分担を変更すれば、キューが満杯又は空になることはほとんど生じなくなる。



【特許請求の範囲】

【請求項1】少なくとも2つのグループに分けることができる複数の処理装置を有するコンピュータにおいて、各前記グループにおける、第1段から第n段までの一連の処理の分担を動的に変更する方法であって、

前記グループ間の処理結果の受渡しのためのキューにおける特徴量の増減を検出するステップと、

前記特徴量の増減に基づき、各前記グループにおける処理の分担を変更する変更ステップと、

を含む処理分担動的変更方法。

【請求項2】前記特徴量は前記キューに格納された情報の量である、請求項1記載の処理分担動的変更方法。

【請求項3】前記特徴量は、グラフィックスに関連する処理の場合、前記キューに格納された頂点データの数である、請求項1記載の処理分担動的変更方法。

【請求項4】前記変更ステップは、

前記特徴量が所定のしきい値以上増加した場合には、前記グループ間の処理分担の境界である第i段 ($1 \leq i < n$) までの処理を実施していたグループの担当を第i段より後段の処理まで当該グループの担当と設定するステップ、

を含む請求項1記載の処理分担動的変更方法。

【請求項5】前記変更ステップは、

前記特徴量が所定のしきい値以上減少した場合には、前記グループ間の処理分担の境界である第i段 ($1 < i \leq n$) までの処理を実施していたグループの担当を第i段より前段の処理までを当該グループの担当と設定するステップ、

を含む請求項1記載の処理分担動的変更方法。

【請求項6】前記処理結果は、何段目までの処理を実施したかについての情報を含む、請求項1記載の処理分担動的変更方法。

【請求項7】前記キューの使用量が上限に達していないか検査するステップと、

前記キューの使用量が上限に達している場合、前記グループ間の処理分担の境界である第i段 ($1 \leq i < n$) までの処理を実施しているグループに属する処理装置が前記キューの最後尾の処理結果を取り出し、前記第i段より後段の処理まで実施した後に、前記キューに当該処理結果を格納するステップと、

をさらに含む請求項1記載の処理分担動的変更方法。

【請求項8】少なくとも2つのグループに分けることができ、各前記グループにおける、第1段から第n段までの一連の処理の分担が設定される、複数の処理装置と、前記グループ間の処理結果の受渡しのためのキューと、前記キューにおける特徴量の増減を検出し、前記特徴量の増減に基づき、各前記グループにおける処理の分担を変更するコントローラと、を有するコンピュータ。

【請求項9】前記コントローラは、

前記特徴量が所定のしきい値以上増加した場合には、前記グループ間の処理分担の境界である第i段 ($1 \leq i < n$) までの処理を実施していたグループの担当を第i段より後段の処理まで当該グループの担当と設定する、請求項8記載のコンピュータ。

【請求項10】前記コントローラは、

前記特徴量が所定のしきい値以上減少した場合には、前記グループ間の処理分担の境界である第i段 ($1 < i \leq n$) までの処理を実施していたグループの担当を第i段より前段の処理までを当該グループの担当と設定する、請求項8記載のコンピュータ。

【請求項11】前記コントローラは、

前記キューの使用量が上限に達していないか検査し、前記キューの使用量が上限に達している場合、前記グループ間の処理分担の境界である第i段 ($1 \leq i < n$) までの処理を実施しているグループに属する処理装置に、前記キューの最後尾の処理結果を取り出し、前記第i段より後段の処理まで実施した後に、前記キューに当該処理結果を格納するように命ずる、請求項8記載のコンピュータ。

【請求項12】少なくとも2つのグループに分けることができる複数の処理装置を有するコンピュータに、各前記グループにおける、第1段から第n段までの一連の処理の分担を動的に変更させるプログラムを格納した記憶装置であって、

前記プログラムは、前記コンピュータに、前記グループ間の処理結果の受渡しのためのキューにおける特徴量の増減を検出するステップと、前記特徴量の増減に基づき、各前記グループにおける処理の分担を変更する変更ステップと、を実行させる、記憶媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、マルチプロセッサ・システムにおける処理分担の動的変更に関し、より詳しくは、コンピュータにおけるホスト・プロセッサとグラフィックス・アダプタにおける処理分担の動的変更に関する。

【0002】

【従来の技術】OpenGLやDirect3Dといったポリゴンベースの三次元グラフィックスの場合、全体のパフォーマンスを決める主な要因は以下のようなものがある。

(1) API

アプリケーションからAPIを介して描画コマンドを発行する速度

(2) ジオメトリ (Geometry) 処理

三角形分割/座標変換/照度計算といったジオメトリ処理の速度

(3) セットアップ (Setup) 処理

三角形の辺/面に沿った色/Z座標値/テクスチャ座標値の勾配計算の速度

(4) ラスタ処理

三角形内部のピクセルの色/Z座標値/テクスチャ座標値を補間して求め、それらをフレーム・バッファへ読み書きする速度

【0003】(1)は、最も処理量が多い方法であるところの、頂点ごとにAPIを呼び出す方法を用いても、一頂点当たり数十クロックしかかからないので、(1)が問題となることはない。

【0004】(4)は一秒間に何ピクセル描画できるか(ピクセル・レート(pixel rate)と呼ぶ)に相当する。このピクセル・レートはポリゴン・レート(polygon rate、後述)とは無関係であり、画面サイズ(例えば640×480や1024×768といった一画面が何ピクセルから構成されるか。)とフレーム・レート(frame rate、一秒間に何コマ表示を行うか。CRTのリフレッシュ・レート(Refresh rate)とは異なる。)と画面上での平均的な重なり(通常3回程度)によって要求量が決定されるものである。最近のグラフィックス・アダプタではSXGA(1280×1024ピクセル)程度の画面サイズまで、ほぼ問題ない程度になりつつある。

【0005】(2)及び(3)のジオメトリとセットアップ処理のパフォーマンスは直接一秒間に何ポリゴン処理できるか(先に述べたポリゴン・レート)に相当する。セットアップ処理は、ジオメトリ処理の一部とされることも多いので、ここではジオメトリ処理として扱うことにする。ジオメトリ処理は数多くの浮動小数点演算が必要である。これには1頂点あたりの処理に数百から数千クロックかかる。従って、ホスト・プロセッサの処理能力だけでは不十分な場合が多い。例えば、10M個の頂点を1秒間に処理する場合、1頂点処理するのに1000クロック必要ならば、10Gクロック/秒のプロセッサが必要になってしまう。よって、グラフィックス・アダプタ上にジオメトリ処理専用の演算器を設けることが多い。また処理の条件、例えば光源の数や光源の種類等によって処理量が大きく変化する。

【0006】ところで、ホスト・プロセッサは描画コマンド(Graphics Command)の列を主記憶装置に蓄える。この描画コマンドの列をコマンド・キュー(Command Queue)という。グラフィックス・アダプタはDMAを用いてコマンド・キューの内容を内部に取り込み、処理を施して表示装置に表示する。このコマンド・キューはDMA転送を行う必要上物理的に主記憶装置内に又はグラフィックス・アダプタ上に存在しなければならない。従って、コマンド・キューの大きさには制限がある。このコマンド・キューが処理の途中で満杯になったり、空になったりすると、ホスト・プロセッサ又はグラフィックス・アダプタが停止するので、全体としてのパフォーマ

ンスが落ちてしまう。コマンド・キューが満杯(Full)の場合には、ホスト・プロセッサはこれ以上コマンド・キューに書き込むことができないので、それに空きが生じるまでホスト・プロセッサは処理を進めることができない。また、コマンド・キューが空(Empty)である場合には、グラフィックス・アダプタは処理を行うことができない。

【0007】ホスト・プロセッサの処理速度とグラフィックス・アダプタの処理速度が等しければコマンド・キューは満杯又は空になることはないが、以下の理由で両者の処理速度を均等にすることはできなかった。

(a)表示のために使用できるホスト・プロセッサの処理能力が予測困難である。ホスト・プロセッサの種類/動作周波数/個数が多様である。また、表示以外で使用するホスト・プロセッサの負荷が予測困難あり、動的に変化する。

(b)上述のジオメトリ処理のように、描画コマンドのホスト・プロセッサ上での処理量(ワークロード:work load)は、現在の状態やデータに依存して動的に変化する。予測困難である。例えばクリッピング(clipping)によって頂点の数は増減する。

(c)描画コマンドのグラフィックス・アダプタ上での処理量は、現在の状態やデータに依存して動的に変化する。予測困難である。

【0008】ホスト・プロセッサの処理能力と処理量をそれぞれ P_h 、 L_h とし、グラフィックス・アダプタの処理能力と処理量をそれぞれ P_a 、 L_a とすると、 $L_h/P_h = L_a/P_a$ が成り立てば、コマンド・キューが空になったり満杯になったりすることなく処理が進められるが、 L_h 、 P_h 、 L_a 及び P_a はいずれも予測不可能であり、必ずしもシステムの性能を最大限引き出すことができていなかった。

【0009】特開平2-275581号公報は、機能毎にその機能を使用する際に必要な時間が予め分かっている場合に、利用者が使用する幾つかの機能をオン/オフすることにより、複数のプロセッサの処理分担を変更して、システム全体の処理速度を向上させる技術を開示している。しかし、機能を使用する際に必要な時間は、処理するデータに依存するため、この公報の前提では処理分担を適切に変更できない。また、ホスト・プロセッサはマルチタスクOSの環境にあることが多く、グラフィックスに割り当てられる計算能力は時々刻々と変化する。ため、この点においてもこの公報の前提は適切ではない。さらに、この公報では、機能のオン/オフの全ての組み合わせに対応する処理分担表を作っておく必要があるが、実際の環境においてはオン/オフできる機能の数は膨大になり实际的ではない。

【0010】

【発明が解決しようとする課題】よって、本発明では、 L_h 、 P_h 、 L_a 及び P_a がいずれも予測不可能である環境

において、 $L_h/P_h = L_o/P_o$ に近づけることができるようなコンピュータ・システムを提供することである。

【0011】また、 $L_h/P_h = L_o/P_o$ に近づけることにより、システム全体のパフォーマンスをできる限り引き出すことができるようにすることも目的である。

【0012】さらに、将来のホスト・プロセッサの処理能力の向上に適應できるようにし、製品寿命を伸ばすことも目的である。

【0013】さらに、コマンド・キューが満杯になるような事態が生じて、ホスト・プロセッサの処理を停止せずにシステム全体のパフォーマンスを下げないようにすることも目的である。

【0014】

【課題を解決するための手段】少なくとも2つのグループに分けることができる複数の処理装置を有するコンピュータにおいて、各グループにおける、第1段から第n段までの一連の処理の分担を動的に変更する本発明は、グループ間で処理結果を受渡すためのキューにおける特徴量の増減を検出するステップと、特徴量の増減に基づき、各グループにおける処理の分担を変更する変更ステップとを含む。キューに格納されたデータの特徴量は、処理量に関係する値を表すものであり、この特徴量を参照しつつ処理分担を変更すれば、キューが満杯又は空になることはほとんど生じなくなる。

【0015】例えば、この特徴量は、キューに格納された情報の量又はキューのサイズ（長さ）であっても、グラフィックスに関連する処理の場合、キューに格納された頂点データの数であってもよい。

【0016】また、先に述べた変更ステップは、特徴量が所定のしきい値以上増加した場合には、グループ間の処理分担の境界である第i段（ $1 \leq i < n$ ）までの処理を実施していたグループの担当を第i段より後段の処理まで当該グループの担当と設定するステップを含むようにすることも考えられる。後段の処理とは、第i+1段以降の処理である。また、特徴量が所定のしきい値以上減少した場合には、グループ間の処理分担の境界である第i段（ $1 < i \leq n$ ）までの処理を実施していたグループの担当を第i段より前段の処理までを当該グループの担当と設定するステップを実行するようにすることも考えられる。また、前段の処理とは、第i-1段以前の処理である。

【0017】またこのように処理分担を動的に変えると、後段の処理を実施するグループは、その段階から処理を実施すべきかという情報が必要になる場合がある。よって、処理結果は、何段目までの処理を実施したかについての情報を含むようにすることも考えられる。

【0018】また、キューの使用量が上限に達していないか検査するステップと、キューの使用量が上限に達している場合、グループ間の処理分担の境界である第i段（ $1 \leq i < n$ ）までの処理を実施しているグループに属

する処理装置がキューの最後尾の処理結果を取り出し、第i段より後段の処理まで実施した後に、キューに当該処理結果を格納するステップをさらに含むようにすることも考えられる。これにより、コマンド・キューが満杯になるような事態が生じて、ホスト・プロセッサの処理を停止せずにシステム全体のパフォーマンスを下げないようにすることができる。

【0019】本発明を実施したコンピュータは、少なくとも2つのグループに分けることができ、各前記グループにおける、第1段から第n段までの一連の処理の分担が設定される、複数の処理装置と、グループ間の処理結果の受渡しのためのキューと、キューにおける特徴量の増減を検出し、特徴量の増減に基づき、各グループにおける処理の分担を変更するコントローラとを有する。

【0020】以上本発明の構成を説明したが、各ステップを実行するようなプログラムにて本発明を実施することも可能である。その際、プログラムはCD-ROMやフロッピー・ディスク等の記憶媒体に記憶されたり、ハードディスクやROM等の記憶装置又はデバイスに記憶される場合がある。また、本発明の処理を実施するような専用の回路又は装置を実施することも可能である。

【0021】

【発明の実施の形態】図1にバイブライン式に処理すべきプロセス群を示す。図1では説明を簡単にするため、プロセス0乃至4を示しているが、これより多いプロセスが存在する場合及び少ないプロセスが存在する場合の両方が考えられる。グラフィックスに関する処理もこのようなバイブライン式に実施する必要がある。

【0022】図1のようなプロセス0乃至4を3つの処理装置グループで処理する場合の一例を図2に示している。処理装置グループ1（1）はプロセス0及び1を担当しており、処理装置グループ2（3）はプロセス2及び3を担当しており、処理装置グループ3（5）はプロセス4を担当している。処理装置グループとしているのは、各処理装置グループに含まれるプロセッサの数は1つでも複数でもよいからである。この処理装置グループ間のデータの交換にはコマンド・キュー1（7）及びコマンド・キュー2（9）が用いられる。

【0023】コマンド・キュー1（7）は処理装置グループ1及び2の間のデータの交換に、コマンド・キュー2（9）は処理装置グループ2及び3の間のデータの交換に用いられる。このコマンド・キューは、処理装置グループが用いるメモリの一部に設けられるようにしても、別個設けるようにしてもよい。コマンド・キューは、コマンド・キューを挟んだ両側の処理装置グループの処理速度が多少変動しても、ある程度の時間空になったり満杯になったりしないような大きさが必要である。これは、処理装置グループの他のプロセスやスレッドにより処理能力が変動したり、両側の処理装置グループの処理量は離散的にしか分割できないため、正確には、

$P_1 = L_2 / P_2$ (1及び2は処理装置グループ1及び2を示す)を保つことはできないからである。

【0024】また、処理分担コントローラ11は、コマンド・キュー1及び2を監視して、処理分担を変更する際の指標となる特徴量の増減を検出する。処理分担コントローラ11は必要に応じて処理分担を変更するよう処理装置グループに命ずる。なお、図2では1つの処理分担コントローラ11を設けているが、各処理装置グループ内で同様の機能を実施するようにすることも可能である。

【0025】処理装置グループ1(1)は、プロセス0及びプロセス1を実行し、コマンド・キュー1(7)にプロセス1の処理結果を順次書き込み、例えばDMA(図示せず)のような機構を介してプロセス2を実施する処理装置グループ2(3)にデータを渡す。処理装置グループ2(3)は、コマンド・キュー1(7)における処理結果を順次用いてプロセス2及びプロセス3を実施し、コマンド・キュー2(7)にプロセス3の処理結果を書き込み、同じようにしてプロセス4を実施する処理装置グループ3(5)にデータを渡す。処理装置グループ3はコマンド・キュー2(9)における処理結果を順次用いてプロセス4を実施する。

【0026】図3に、コマンド・キュー1(7)における特徴量が所定のしきい値以上減少した場合を示す。コマンド・キュー1(7)における特徴量が所定のしきい値以上減少した場合には、処理装置グループ1(1)の負荷が大きくなったか又は処理能力が落ちたか、若しくは処理装置グループ2(3)の負荷が軽くなったか又は処理能力が上がったか、である。とにかく、このまま放置するとコマンド・キュー1(7)が空になって、処理装置グループ2(3)が遊んでしまう。そこで、処理分担コントローラ11は、処理装置グループ1にプロセス0のみ実行するように命じ、処理装置グループ2(3)にプロセス1も実行するように命ずる。図3はこのような処理分担の変更を実施した後の状態を示している。もし、この後コマンド・キュー2(9)における特徴量が所定のしきい値以上減少するような場合には、再度処理装置グループ2(3)及び処理装置グループ3(5)の間で担当プロセスの調整がなされる。

【0027】通常のコンピュータにおけるグラフィックス処理を考えた場合には、図2及び図3における処理装置グループは2つで、ホスト・プロセッサ(1又は複数)のグループとグラフィックス・アダプタのグループに分けられる。図4に本発明のコンピュータの例を示す。ホスト・プロセッサ21はメモリ・コントローラ23に接続している。メモリ・コントローラ23はメイン・メモリ25及びバス27に接続している。バス27にはグラフィックス・アダプタ29が接続しており、このグラフィックス・アダプタ29は表示装置31に接続している。ホスト・プロセッサ21は上でも述べているよ

うに複数のプロセッサであってもよい。メモリ・コントローラ23にはDMAコントローラを含む。また、メイン・メモリ25内には、コマンド・キュー25aと、ソフトウェアとして実施されている処理分担コントローラ25cと、本発明ではホスト・プロセッサ21が処理した頂点データの数(コマンド・キュー25aへの入力となる)をカウントするカウンタ1(25b)とを含む。また、メイン・メモリ25には実行中の他のソフトウェア(割り当てられたプロセスに関するプログラムを含む)も含まれ、それらは必要に応じて図示しないハードディスク等からロードされる。グラフィックス・アダプタ29には、図示しないジオメトリ・プロセッサ及びラスタ・プロセッサと、グラフィックス・アダプタ29が処理した頂点データの数(コマンド・キュー25aへの入力となる)をカウントするカウンタ29aと、割り当てられる可能性のある処理プロセスに必要なプログラム(図示せず)を含む。

【0028】ホスト・プロセッサ21は、図示しないプログラムを用いて割り当てられたプロセスの処理を実施し、その結果をコマンド・キュー25aにメモリ・コントローラ23を介して書き込む。頂点データを1つ処理するごとにカウンタ1(25b)を1インクリメントする。また、メモリ・コントローラ23は所定のサイズごとにコマンド・キュー25aのデータをバス27を介してグラフィックス・アダプタ29に渡す。グラフィックス・アダプタ29は渡されたデータ(ホスト・プロセッサ21の処理結果)を用いて割り当てられたプロセスを実施し、処理結果として表示装置31にグラフィックスを表示する。なお、グラフィックス・アダプタ29において頂点データが1つ処理されるごとにカウンタ2(29a)を1インクリメントする。

【0029】ホスト・プロセッサ21側に設けられた処理分担コントローラ25cは、一定期間ごとにカウンタ1(25b)及びカウンタ2(29a)の値を取得し、各カウンタ値の増分を用いて後に説明する処理分担の変更を行う。本発明では、ホスト・プロセッサ21側に処理分担コントローラ25cを設けているが、これはグラフィックス・アダプタ29側に設けることも可能である。

【0030】コマンド・キュー25aの一例を図5に示す。コマンド・キュー25aは、DMA時のページ・マッピングの問題や、ホスト・プロセッサ21が複数のプロセッサを含むような場合のロックの問題から、ページ境界(page boundary)に沿ったページ・サイズ以下のリンク・リスト(Linked List)構造が好ましい。リンク・バッファはポインタを格納しており、各ポインタの先には各キュー・エレメントが配置されている。キュー・エレメントは描画コマンド(Command)及びデータ(Data)を含んでおり、ページ・サイズ以下で例えば2-4Kバイトである。書き込みポインタ(Write PTR)はホスト・プロセッサ21が書き込ん

だキュー内の最後のキュー・エレメントを指す。読み出しポインタ (Read PTR) はグラフィックス・アダプタ29が次に読み出すべきキュー・エレメントのアドレスを指す。DMAでキュー・エレメントを読み出すと、読み出しポインタを1つ進める。

【0031】ホスト・プロセッサ21とグラフィックス・アダプタ29の間で処理プロセス間のデータの受渡し方法を予め定義しておき、ホスト・プロセッサ21がコマンド・キュー25aに処理結果として描画コマンドとデータを渡す際に、どの処理プロセスまで実施したかを表すタグを付加しておく。タグには、例えばプロセス2まで実施したということを含めてもよいし、プロセス3から実施しろということを含めても良い。このタグは例えばキュー・エレメント内の描画コマンド内に含める。このようにすれば、グラフィックス・アダプタ29がどの処理プロセスから残りの処理プロセスを実施すればよいかが分かるので、全体として正しい結果が得られる。

【0032】処理分担コントローラ25cの処理を説明する前に、処理分担コントローラ25cが処理分担を変更するために参照する特徴量について説明しておく。図4に示したように、本実施例では頂点データの数を特徴量としている。これは、コマンド・キュー25aには頂点データ以外にもビットマップ等の二次元イメージを描画する命令も含まれる。この場合にはジオメトリ処理は必要なく素通りし、ラスト処理部で処理される。ビットマップ等がコマンド・キュー25aに入っている場合には、コマンド・キューの大きさは大きくなるし、頂点データに比して処理負荷は軽くなるので、長さ又は大きさを基準にすると、グラフィックス・アダプタ29への処理分担は不正確になるおそれがある。よって、本実施例ではコマンド・キュー25aに含まれる頂点データの数を特徴量として取り扱う。但し、他の状況においてはコマンド・キューの長さや大きさを特徴量とすることが可能な場合も生じえる。

【0033】特徴量である頂点データの数を把握するために、コマンド・キュー25aを走査して、カウントすることも可能である。しかし、本実施例では、ホスト・プロセッサ21とグラフィックス・アダプタ29がそれぞれ自分で処理した頂点データを数を、カウンタ1 (25b) 及びカウンタ2 (29a) でカウントし (それぞれカウンタ1及びカウンタ2とする)、一定期間における増分 (それぞれ Δ カウンタ1及び Δ カウンタ2とする) の差によりコマンド・キュー25a内に存在する頂点データの数を把握することにする。

【0034】例えば、 Δ カウンタ2が所定のしきい値以上 Δ カウンタ1より大きい場合には、グラフィックス・アダプタ29の処理の方が速いので、ホスト・プロセッサ21に割り当てられている処理プロセスをグラフィックス・アダプタ29に割り当てないと、コマンド・キュー25aが空になってしまうと判断できる。また、 Δ カ

ウント1が所定のしきい値以上 Δ カウンタ2より大きい場合には、グラフィックス・アダプタ29の処理の方が遅いので、グラフィックス・アダプタ29に割り当てられた処理プロセスをホスト・プロセッサ21に割り当てないと、コマンド・キュー25cが満杯になっていしまうと判断できる。このような判断を、 Δ カウンタ1及び Δ カウンタ2の関数を定義して行うことも出来る。

【0035】処理分担コントローラ25cの処理フローを図6に示す。まず、特徴量の増減を検出するために、カウンタ1及びカウンタ2の値を取得し、 Δ カウンタ1及び Δ カウンタ2を計算する (ステップ110)。そして、 Δ カウンタ1が Δ カウンタ2より第1のしきい値以上大きいと判断する (ステップ120)。もし大きい場合には、コマンド・キュー25a内の頂点データは許容できる範囲を超えて増加しているため、満杯になってしまいう可能性がある。よって、グラフィックス・アダプタ29の担当処理プロセスを第i段以降から第i+p段以降に変更する (ステップ150)。ここでpは正の整数である。なお、グラフィックス・アダプタ29の担当処理プロセスを減らせば、自動的にホスト・プロセッサ21の担当分が増加する。

【0036】一方、ステップ120の条件が満たされない場合には、 Δ カウンタ2が Δ カウンタ1より第2のしきい値以上大きいと判断する (ステップ130)。もし、この条件が満たされる場合には、コマンド・キュー25a内の頂点データ数が許容範囲を超えて減少しているため、コマンド・キュー25aは空になってしまうかもしれない。そこで、ホスト・プロセッサ21の担当処理プロセスを第i段以前から第i-m段 (mは正の整数) 以前に変更する (ステップ140)。

【0037】そして、所定時間経過後、再度ステップ110からの処理を繰り返す。

【0038】このようにして、特徴量の増減に基づき、各処理装置グループにおける処理の分担を変更する。

【0039】図6の処理フローは様々な変更が可能である。 Δ カウンタ1及び Δ カウンタ2の差である Δ カウンタの増減によって、処理分担を変更するようにしてもよい。また、第1及び第2のしきい値は同じでも異なっても良い。また、p及びmは固定であっても Δ カウンタ1及び Δ カウンタ2の差の大きさによって変化させてもよい。

【0040】ステップ160における所定時間は、コマンド・キュー25aを空にしないような周期、具体的には以下のような条件で決めることができる。すなわち、現在のコマンド・キュー25aの内容を全てグラフィックス・アダプタ29に転送するのに必要な時間をXとし、一つのキュー・エレメントをホスト側にて最高速度 (最大限グラフィックス・アダプタ29に処理を実行させる場合) で作成する時間をCとした場合、処理分担を変更する周期である、この所定時間Tは $T < X - C$ であ

ればよい。

【0041】なお、ホスト・プロセッサ21に処理すべき描画コマンドが残っているのに、コマンド・キュー25c（サイズは通常数十Kバイトから数Mバイト）の使用量が上限に達してしまうと、ホスト・プロセッサ21は処理を続行することができなくなる。よって、例えば図6におけるステップ120のようなタイミングで、コマンド・キュー25aが満杯かどうかを検査して、もし満杯であれば、処理分担コントローラ25cはホスト・コンピュータ21に取り戻し命令を出力し、図7のような処理が実施される。タイミングは他の場合であってもよく、ホスト・プロセッサ21が書き込みの際に自分で判断して図7のような処理を開始しても良い。

【0042】まず、ホスト・コンピュータ21は、コマンド・キュー25a内の最後尾の処理結果を取り出す（ステップ210）。これは、図8のように、リング・バッファの元の書き込みポイント（WRITE_POINT）から最後尾のキュー・エレメント7を取り戻すものである。なお、書き込みポイントの位置を一つ戻す。本実施例では最後尾の1つのキュー・エレメントを取り出すことにしているが、複数のキュー・エレメントを取り出すことにしてもよい。そして、グラフィックス・アダプタ29担当の処理プロセスの一部を、取り戻した処理結果に対して実施する（ステップ230）。例えば第i段まで担当していた場合には、第i+1段以降を実施する。ホスト・プロセッサ21はどの処理プロセスまで実施したかはキュー・エレメントの描画コマンド部分に付されたタグを見れば分かるので、その後の段階の処理プロセスを実施する。図8の例では、後段の処理プロセスを実施した結果が、キュー・エレメント7'として示されている。どの段階まで処理プロセスを実施するかは任意である。但し、場合によっては、再度キュー・エレメント7'を取り戻して処理する場合も生じえる。最後に、実施終了した処理プロセスのタグを付して、再度処理結果をコマンド・キュー25aに入力する（ステップ230）。このタグは、次にグラフィックス・アダプタ29が開始すべき処理プロセスに関する情報でもよい。図9にステップ230終了後の状態を示す。この図ではまだキュー・エレメント0がグラフィックス・アダプタ29へ転送されていないので、コマンド・キュー25aが満杯ということになってしまうが、キュー・エレメント0の転送が終了していれば、ホスト・プロセッサ21は次の処理に取りかかることができる。以上のようにして、コマンド・キューが万が一満杯になった場合に対処する。

【0043】以上本発明の一実施例を説明したが、本発明は上の実施例に限定されない。例えば、コマンド・キューの構造は図5のような構造に限定されない。コマンド・キューはキュー・エレメントをチェーン化して構成することも可能である。また、処理装置グループ内の複

数のプロセッサは同じ能力でなくともよい。また、処理分担コントローラはソフトウェアで実施しても専用の回路などを用意しても良い。

【0044】

【効果】 L_1 、 P_1 、 L_2 及び P_2 （それぞれ第1処理装置グループの処理量、処理能力、第2処理装置グループの処理量、処理能力）がいずれも予測不可能である環境において、 $L_1/P_1=L_2/P_2$ に近づけることができるようなコンピュータ・システムを提供することができた。

【0045】また、 $L_1/P_1=L_2/P_2$ に近づけることにより、システム全体のパフォーマンスをできる限り引き出すことができるようにすることもできた。

【0046】さらに、将来のホスト・プロセッサの処理能力の向上に適應できるようにし、製品寿命を伸ばすこともできた。

【0047】さらに、コマンド・キューが満杯になるような事態が生じて、コマンド・キューに書き込みを行う処理装置グループの処理を停止せずにシステム全体のパフォーマンスを下げないようにすることもできた。

【図面の簡単な説明】

【図1】パイプライン式の処理プロセスの例である。

【図2】複数の処理装置グループにおいて処理プロセスを分担する際の機能ブロック図である。

【図3】図2の状態から処理分担を変更した場合の図である。

【図4】グラフィックス処理の処理プロセスをホスト・プロセッサとグラフィックス・アダプタとで分担する場合の機能ブロック図である。

【図5】コマンド・キュー構造の例である。

【図6】処理分担コントローラの処理フローの例である。

【図7】コマンド・キューが満杯になった際のホスト・プロセッサの処理フローの例である。

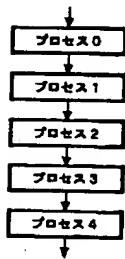
【図8】図7の処理を模式的に示した図である。

【図9】図7の処理結果を模式的に示した図である。

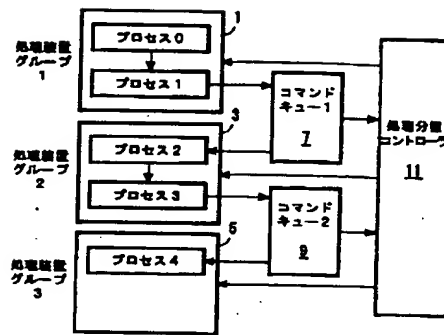
【符号の説明】

- 1、3、5 処理装置グループ
- 7、9 コマンド・キュー
- 11 処理分担コントローラ
- 21 ホスト・プロセッサ
- 23 メモリ・コントローラ
- 25 メイン・メモリ
- 25a コマンド・キュー
- 25b カウンタ1
- 25c 処理分担コントローラ
- 27 バス
- 29 グラフィックス・コントローラ
- 29a カウンタ2
- 31 表示装置

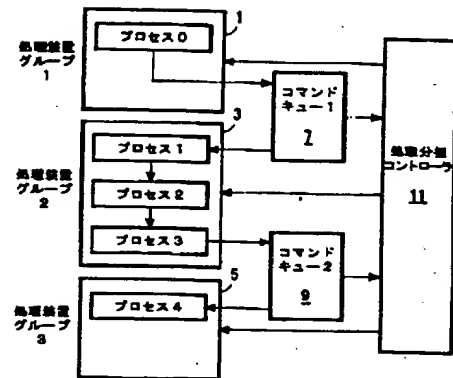
【図1】



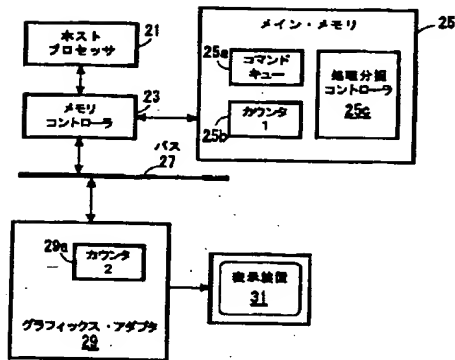
【図2】



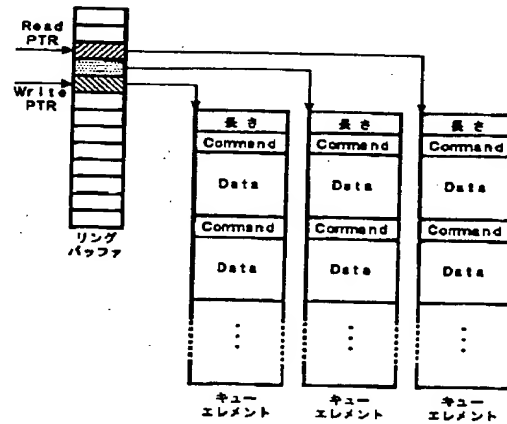
【図3】



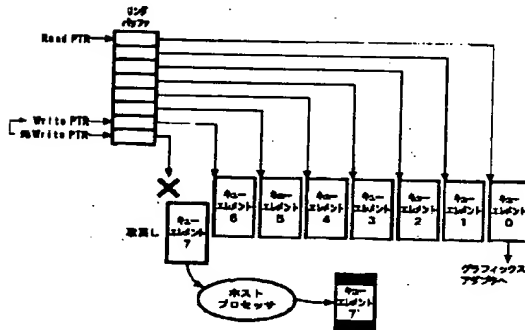
【図4】



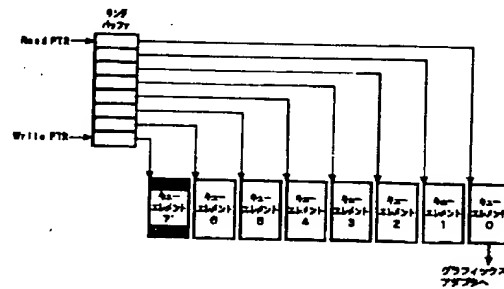
【図5】



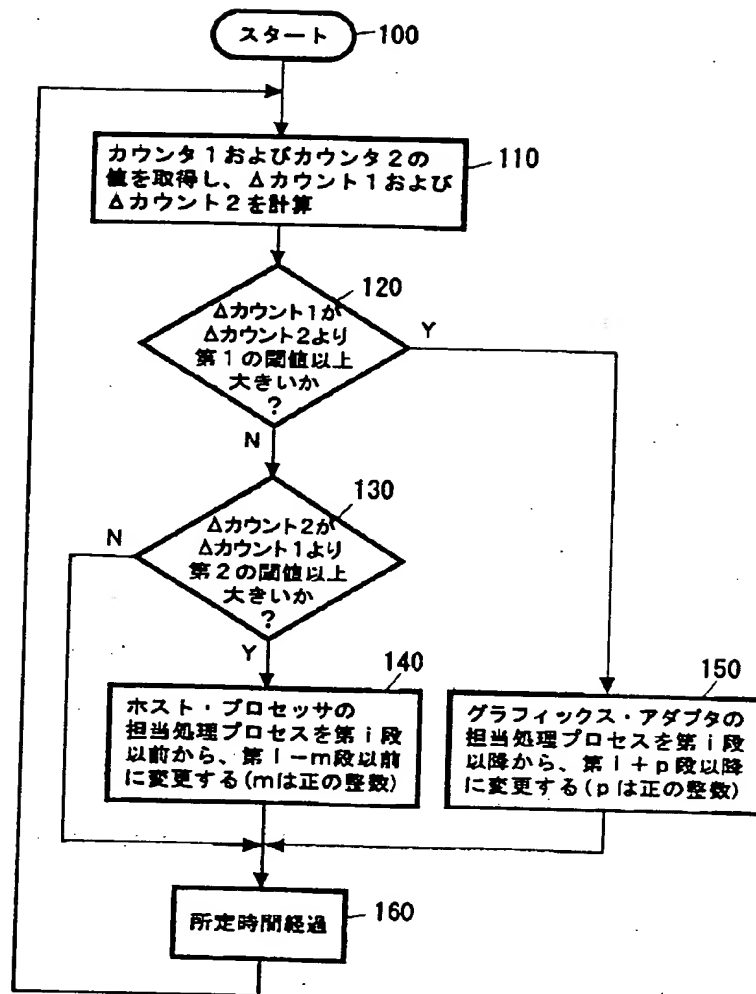
【図8】



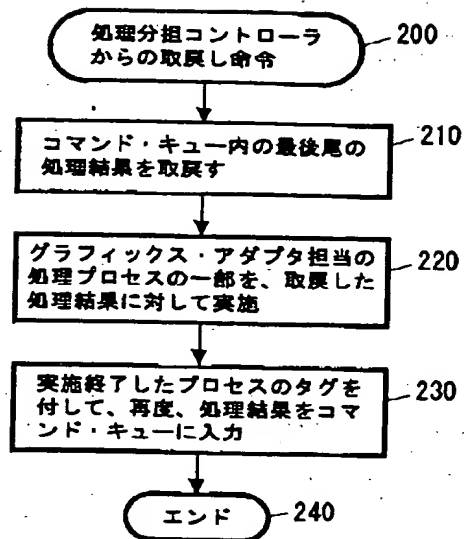
【図9】



【図6】



【図7】



フロントページの続き

(72)発明者 川瀬 桂
神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 東京基礎研究所内

(72)発明者 森山 孝男
神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 東京基礎研究所内

(72)発明者 中村 英史
神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 東京基礎研究所内

Fターム(参考) 5B045 AA01 GG02
5B057 AA01 CH02 CH05